REMARKS

This is intended as a full and complete response to the Office Action dated August 16, 2004, having a shortened statutory period for response set to expire on November 16, 2004. Please reconsider the claims pending in the application for reasons discussed below.

In the specification, the paragraphs [0005] and [0037] have been amended to correct minor editorial problems.

Claims 1-36 are pending in the application. Claims 1-36 remain pending following entry of this response. Applicants submit that the amendments do not introduce new matter.

Claim Rejections - 35 USC § 103(a)

Independent claims 1, 12, 16, 20, and 32 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over *Hicks et al.*, U.S. Patent 6,175,956 (*Hicks*), in view of *Brandes*, U.S. Patent 5,946,484, and further in view of *Percival et al.*, U.S. Patent 6226,652 (*Percival*).

Regarding claims 1 and 20, the Examiner asserts that *Hicks* "discloses the optimization of original source code" and at the same time "does not disclose generating an optimized source code." The Examiner appears to suggest, however, that the combination of *Hicks* in view of *Brandes* discloses the recited limitation from claims 1 and 20; namely "generating an optimized source code for an original source code". Applicants respectfully disagree.

Hicks discloses a computer implemented method and computer program compiler product for implementing method calls in a computer system. *Hicks*, Abstract. More specifically, *Hicks* discloses methods for optimizing object code generated from source code that includes virtual method calls. When software is developed using object oriented programming languages, virtual methods are methods that may be overridden by a subclass. *Hicks*, 1:14-22. Normally when a method is overridden, instances of the subclass will use the overridden method, and instances of the base class will continue to use the original. Virtual methods are one of the central features of

object oriented programming, and they allow programmers to manipulate groups of objects through a standard interface. Virtual methods, however, prevent a compiler from binding a method call to the address of a function during compile time (static binding). Instead, virtual functions must use pointer indirection to resolve an actual method call executed during program execution (dynamic binding). Doing so causes such a computer program to execute more slowly than one that includes only statically bound calls. *Hicks*, 1:29-39.

The methods of *Hicks* provide for optimizing virtual method calls using devirtualization techniques. The optimizations operate on an intermediate representation of source code. *Hicks*, 4:3-33. That is, a compiler first generates an intermediate representation of source code written in an object-oriented programming language. "The optimizer 126 operates on the intermediate representation 134 to generate the optimized or revised representation 136. The code generator 128 converts the revised representation 136 into object code 138 or the actual program that a computer can execute." *Hicks*, 4:16-19.

Applicants, however, do not claim optimizing an intermediate representation. Applicants claim optimizing actual source code, and displaying optimization changes made to source code. To establish *prima facie* obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). "All words in a claim must be considered in judging the patentability of that claim against the prior art." *In re Wilson*, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970). See also MPEP § 2143.03. Since *Hicks* fails to teach, show, or suggest generating an optimized source code for an original source code, Applicants submit that *prima facie* obviousness of the rejected claims has not been established. Therefore, on the basis of *Hicks* alone Applicants respectfully submit that the rejection is improper and should be withdrawn.

The Examiner further asserts *Brandes* discloses generating source code from object code and that in combination with *Percival* (and *Hicks*), the references disclose generating an optimized source code for an original source code and displaying a

change between the optimized source code for the original source code. Respectfully, Applicants submit that a person skilled in the art would not be motivated to combine *Brandes* and *Percival* as suggested by the Examiner.

Brandes operates by generating source code for common patterns of object code. *Brandes*, Abstract. However, the relationship between source code and object is not one-to-one. Thus, many different versions of source code will compile to an identical sequence of machine instructions (i.e., object code) and vice-versa. As a result, decompiling given object code to produce source code will not necessarily produce the original source code that produced the given object code. In other words, the source code generated by *Brandes* has no necessary relationship to the original source code and, therefore, the generated source code would bear little, if any, resemblance to the original source code. Further, as is well known, source code will typically include a number of arbitrary comments, variable names, procedure call names, method names, defined names (e.g., #define statements in C++), and the like. The techniques disclosed by *Brandes*, however, fail to provide any connection between the source code it generates and the original source code. Without a correspondence between the optimized and original source code, the differences therebetween would be so substantial as to render a comparison between the two meaningless. Therefore, a person skilled in the art would not be motivated to combine *Brandes* and *Percival* as suggested by the Examiner. Thus, *Brandes* (together with *Hicks*), in view of *Percival*, fails to teach, show, show, or suggest generating an optimized source code for an original source code as claimed by Applicants.

Regarding claims 12, 16, and 32, each claim recites the limitation of "optimizing the object code to produce an optimized object code." *Hicks* discloses optimizing an intermediate (or revised) representation of source code by a parser (*Hicks*, 4:14-16) and a code generator "that converts the revised representation into object code. *Hicks*, 4:16-18. In contrast, Applicants' claim optimizing the object code to produce an optimized object code. To establish *prima facie* obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). "All words in a claim must be

considered in judging the patentability of that claim against the prior art." *In re Wilson*, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970). See also MPEP § 2143.03. Since *Hicks* does not teach optimizing the object code to produce an optimized object code, as claimed, a *prima facie* case of obviousness has not been established.

Regarding claims 2-11, 13-15, 17-19, 21-31, and 33-36, each of these claims depends from one of independent claims 1, 12, 16, 20 or 32. If an independent claim is non-obvious under 35 U.S.C. § 103, then any claim depending therefrom is non-obvious. *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988). The rejection to these claims, therefore, is obviated by the remarks above regarding independent claims 1, 12, 16, 20, and 32.

Conclusion

Having addressed all issues set out in the office action, Applicants respectfully submit that the claims are in condition for allowance and respectfully request that the claims be allowed.

Respectfully submitted,

Gero G. McClellan
Registration No. 44,227
MOSER, PATTERSON & SHERIDAN, L.L.P.
3040 Post Oak Blvd. Suite 1500
Houston, TX 77056
Telephone: (713) 623-4844
Facsimile: (713) 623-4846
Attorney for Applicants